

Федеральное государственное образовательное бюджетное учреждение
высшего профессионального образования
«Нижегородский Государственный Университет им.
Н.И.Лобачевского» (ННГУ)
Институт Информационных Технологий Математики и Механики

Отчёт по лабораторной работе
Работа с массивами случайно сгенерированных чисел

Выполнил:
студент группы 3821Б1ФИЗ

Мурадов К.Р.

Проверил:
заведующий лабораторией суперкомпьютерных
технологий и высокопроизводительных
вычислений

Лебедев И.Г

Нижний Новгород
2021г.

Содержание

Введение.....	1
Постановка задачи	1
Руководство пользователя	2
Руководство программиста	2
Описание структуры кода программы	2
Описание структуры данных	3
Описание алгоритмов	4
Эксперименты.....	
Заключение	6
Список литературы	6
Приложение 1	7

Введение

Программирование — это интересный, полезный и увлекательный процесс, благодаря которому создаются программы – набор инструкций, которые приводятся в исполнение компьютерами.

Одной из ключевых задач компьютера является работа с данными. В том числе и со случайно генерируемыми наборами данных, о которых пойдет речь в настоящей работе.

Случайные числа — это одна из основных составляющих любого языка программирования, на них строятся многие алгоритмы. Они имеют применение в физике, например, в исследованиях электронного шума, в инженерном деле и исследовании операций. Многие методы статистического анализа требуют использования случайных чисел.

В ходе выполнения лабораторной работы на языке программирования «С» будет написана программа, работающая со случайными числами.

Постановка задачи

Реализовать сортировки массивов данных (тип данных определяется преподавателем) задаваемых: обязательно случайно, дополнительно с клавиатуры или из файла.

Реализовать сортировки: пузырьком, вставкой, быстрая.

Сравнить время работы, сделать выводы.

Первая программа создает текстовый файл с записанными в него числами. Программа принимает количество чисел n , максимальное и минимальное значение.

Вторая программа читает текстовый файл с набором чисел, выводит консольный интерфейс (печать, сортировка, сброс, выход), выполняет выбранные действия.

Дополнительные задания:

1. Первая программа может создавать массив не только из случайных чисел, но и получать их с клавиатуры и из файла.
2. Добавить возможность запуска сортировки через параметры командной строки.
3. Добавить вычисление первой нормы вектора (массива);
4. Добавить вычисление второй нормы вектора (массива);
5. Добавить вычисление Гельдеровой нормы вектора (массива);
6. Добавить вычисление бесконечной нормы вектора (массива);
7. Добавить нормировку вектора (массива);

Руководство пользователя

После запуска программа выводит три сообщения «Введите кол-во случайных чисел:, Введите минимум диапазона, Введите максимум диапазона» означающим, что от пользователя для дальнейшей работы требуется ввести: число элементов массива случайных чисел в количестве n , нижнюю границу генерации чисел \min и верхнюю \max , в одну строку через пробелы, как указано в сообщении от программы (см. рис. 1).

Например, введем $n = 10$, $\min = 5$, $\max = 100$

```
Введите кол-во случайных чисел: 10
Введите минимум диапазона: 5
Введите максимум диапазона: 100
```

Рисунок 1. Терминал после запуска

После нажатия Enter, программа выведет результат суммирования сгенерированных чисел, произведенного по правилам технического задания. На этом программа завершается (см. рис. 2).

```
Введите кол-во случайных чисел: 10
Введите минимум диапазона: 0
Введите максимум диапазона: 99

Полученная сумма: -439.305184
```

Рисунок 2. Результат работы программы



Рисунок 3. Записанные числа в файле.

После завершения первой программы следует запустить вторую. После ее запуска на экран будет выведен консольный интерфейс. (см. рис. 4)

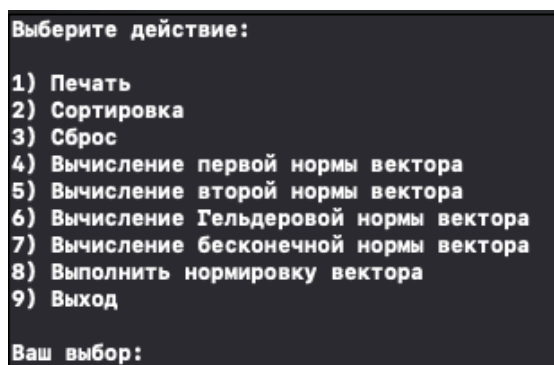


Рисунок 4. Консольный интерфейс.

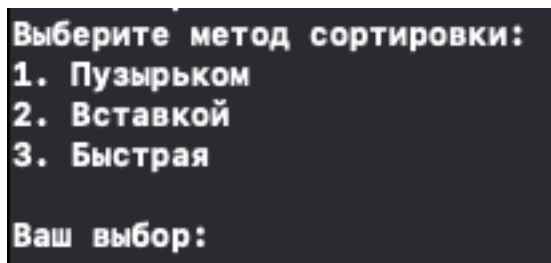
Консольный интерфейс представляет собой набор из девяти команд:

1. Печать
2. Сортировка
3. Сброс
4. Вычисление первой нормы вектора
5. Вычисление второй нормы вектора
6. Вычисление Гельдеровой нормы вектора
7. Вычисление бесконечной нормы вектора
8. Выполнить нормировку вектора
9. Выход

- Команда «Печать» осуществляет вывод на экран чисел, записанных в файле.
- Команда «Сортировка» сортирует числа.
- Команда «Сброс» сбрасывает сортировку.
- Команда «Вычисление первой нормы вектора» вычисляет первую норму вектора чисел.
- Команда «Вычисление второй нормы вектора» вычисляет вторую норму вектора чисел.
- Команда «Вычисление Гельдеровой нормы вектора» вычисляет Гельдерову норму вектора чисел.
- Команда «Вычисление бесконечной нормы вектора» вычисляет бесконечную норму вектора чисел.
- Команда «Выполнить нормировку вектора» выполняет нормировку вектора.
- Команда «Выход» завершает программу.

Чтобы вывести на экран числа, отсортированные по возрастанию, следует сначала выполнить «Сортировку» и после «Печать».

При выборе команды «Сортировка» пользователю предоставляется выбор типа сортировки. (см. рис. 5)

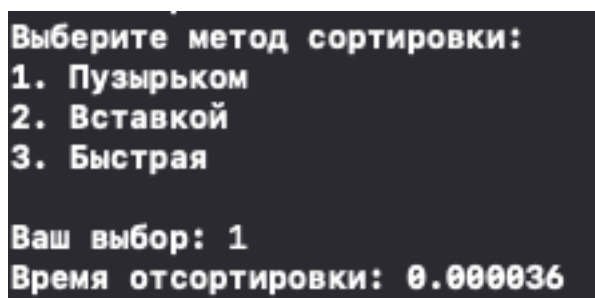


```
Выберите метод сортировки:
1. Пузырьком
2. Вставкой
3. Быстрая

Ваш выбор:
```

Рисунок 5. Типы сортировки.

После выполнения какой-либо сортировки на экран будет выведено сообщение о времени занимаемом этим типом сортировки. (см. рис. 6)



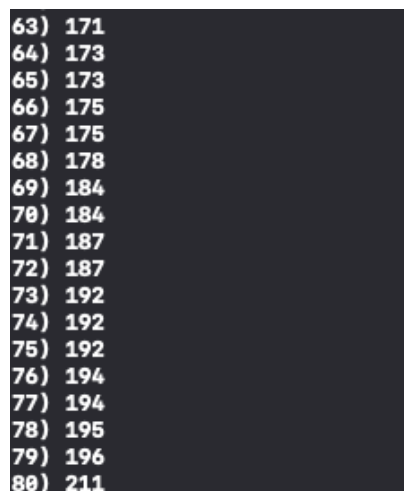
```
Выберите метод сортировки:
1. Пузырьком
2. Вставкой
3. Быстрая

Ваш выбор: 1
Время отсортировки: 0.000036
```

Рисунок 6. Время сортировки.

Чтобы выполнить другую сортировку нужно сбросить предыдущую. Для этого следует воспользоваться командой «Сброс».

Таким образом, мы можем разными способами сортировать числа в файле, сравнивать скорость сортировок и выводить отсортированные по возрастанию числа на экран. (см. рис. 7)



```
63) 171
64) 173
65) 173
66) 175
67) 175
68) 178
69) 184
70) 184
71) 187
72) 187
73) 192
74) 192
75) 192
76) 194
77) 194
78) 195
79) 196
80) 211
```

Рисунок 7. Отсортированные числа

При выборе команды «Вычисление первой нормы вектора» и других команд норм вектора, выводится результат о вычислении (см. рис. 8)

```
Выберите действие:
1) Печать
2) Сортировка
3) Сброс
4) Вычисление первой нормы вектора
5) Вычисление второй нормы вектора
6) Вычисление Гельдеровой нормы вектора
7) Вычисление бесконечной нормы вектора
8) Выполнить нормировку вектора
9) Выход

Ваш выбор: 4
13167.000000
```

Рисунок 8. Результат вычисления вектора

Таким образом мы можем вычислить нормы вектора любых чисел.

Руководство программиста

Описание структуры программы.

Первая программа состоит из одного модуля `int main() {...}`, в котором находится код создания чисел и записи их в файл. Вторая программа также состоит из одного модуля `int main() {...}`, в котором находится код сортировки чисел.

Описание алгоритмов.

1. Алгоритм записи случайных чисел в файл.

```
randomNumber = (unsigned char*)malloc(n * sizeof(unsigned char));
randomNumberDecimal = (int*)malloc(n *
sizeof(int));

// создаем случайное число и узнаем десятичную часть числа
for (i = 0; i < n; i++) {

    randomNumber[i] = RandomFloat(min, max);
    randomNumberDecimal[i] = (randomNumber[i] -
(int)randomNumber[i]) * 1000000;
    fprintf(file, "%hu\n", randomNumber[i]);
}
```

2. Функция генерирования случайных чисел.

```
float RandomNumber(int min, int max){
    return ((max - min) * ((unsigned char)rand() / RAND_MAX)) +
min;
}
```

3. Алгоритм считывания чисел из файла.

```
fp = fopen("textFile.txt", "r");

while((ch=fgetc(fp))!=EOF) {
    if(ch=='\n')
        linesCount++;
}

randomNumber = (unsigned char*)malloc(linesCount *
sizeof(unsigned char));
pRandomNumber = (unsigned char*)malloc(linesCount *
sizeof(unsigned char));

rewind(fp);
```

```

for (i = 0; i < linesCount; i++){
    fscanf(fp, "%hhu", &randomNumber[i]);
}

```

4. Алгоритм консольного интерфейса.

```

do{
    printf("Выберите действие:\n\n1) Печать\n2) Сортировка\n3)
Сброс\n4) Вычисление первой нормы вектора\n5) Вычисление второй
нормы вектора\n6) Вычисление Гельдеровой нормы вектора\n7)
Вычисление бесконечной нормы вектора\n8) Выполнить нормировку
вектора\n9) Выход\n\nВаш выбор: ");
    scanf("%d", &menuOption);
    secondSum = 0;

    switch(menuOption){
        case 1:
            for (i = 0; i < linesCount; i++)
            {
                printf("%d) %hhu\n", i + 1,
randomNumber[i]);
            }
            sleep(2);
            break;

        case 2:
            do{
                printf("Выберите метод сортировки:\n1.
Пузырьком\n2. Вставкой\n3. Быстрая\n\nВаш выбор: ");
                scanf("%d", &sort);
                switch(sort){
                    case 1: // пузырьк
bubbleSort(linesCount, randomNumber, t, fp);
                    break;
                    case 2: // вставка
insertSort(linesCount, randomNumber, t, fp);
                    break;
                    case 3:
fastSort(linesCount, randomNumber, t, fp);
                    break;
                    default:
                        printf("Не правильный выбор,
пожалуйста попробуйте еще раз.\n\n");
                        break;
                }
            }while(sort <= 0 || sort > 3);

```

```

        break;

    case 3:
        for (i = 0; i < linesCount; i++){
            fscanf(fp, "%hhu", &randomNumber[i]);
        }
        printf("Массив сброшен.\n");
        sleep(2);
        break;

    case 4:
        for(i = 0; i < linesCount; i++){
            pRandomNumber[i] = (randomNumber[i]);
            secondSum += pRandomNumber[i];
        }
        printf("%f\n", secondSum);
        sleep(2);
        break;

    case 5:
        for(i = 0; i < linesCount; i++){
            pRandomNumber[i] = (randomNumber[i]);
            secondSum += pRandomNumber[i] *
pRandomNumber[i];
        }

        printf("%f\n", sqrt(secondSum));
        sleep(2);
        break;

    case 6:
        printf("Введите значение P: ");
        scanf("%d", &p);
        for(i = 0; i < linesCount; i++){
            secondSum += pow((randomNumber[i]), p);
        }
        normVector = pow(secondSum, 1.0/p);
        printf("%f\n\n", normVector);
        sleep(2);
        break;

    case 7:
        for(i = 0; i < linesCount; i++){
            pRandomNumber[i] = (randomNumber[i]);
        }
        for(i = 0; i < linesCount; i++){
            if(pRandomNumber[i] > maxVector){
                maxVector = pRandomNumber[i];
            }
        }

```

```

        }
    }
    printf("Бесконечная норма вектора равна %f",
maxVector);
    sleep(2);
    break;

    case 8:
        for(i = 0; i < linesCount; i++){
            pRandomNumber[i] = (randomNumber[i]);
            secondSum += pRandomNumber[i] *
pRandomNumber[i];
        }
        sqrtedNums = sqrt(secondSum);
        for(i = 0; i < linesCount; i++){
            printf("%d) %f\n", i +
1, randomNumber[i]/sqrtedNums);
        }
        sleep(2);
        break;

    case 9:
        printf("Завершение программы...\n\n");
        sleep(3);
        break;

    default:
        printf("Не корректный ввод\n\n");
}

}while(menuOption != 9);

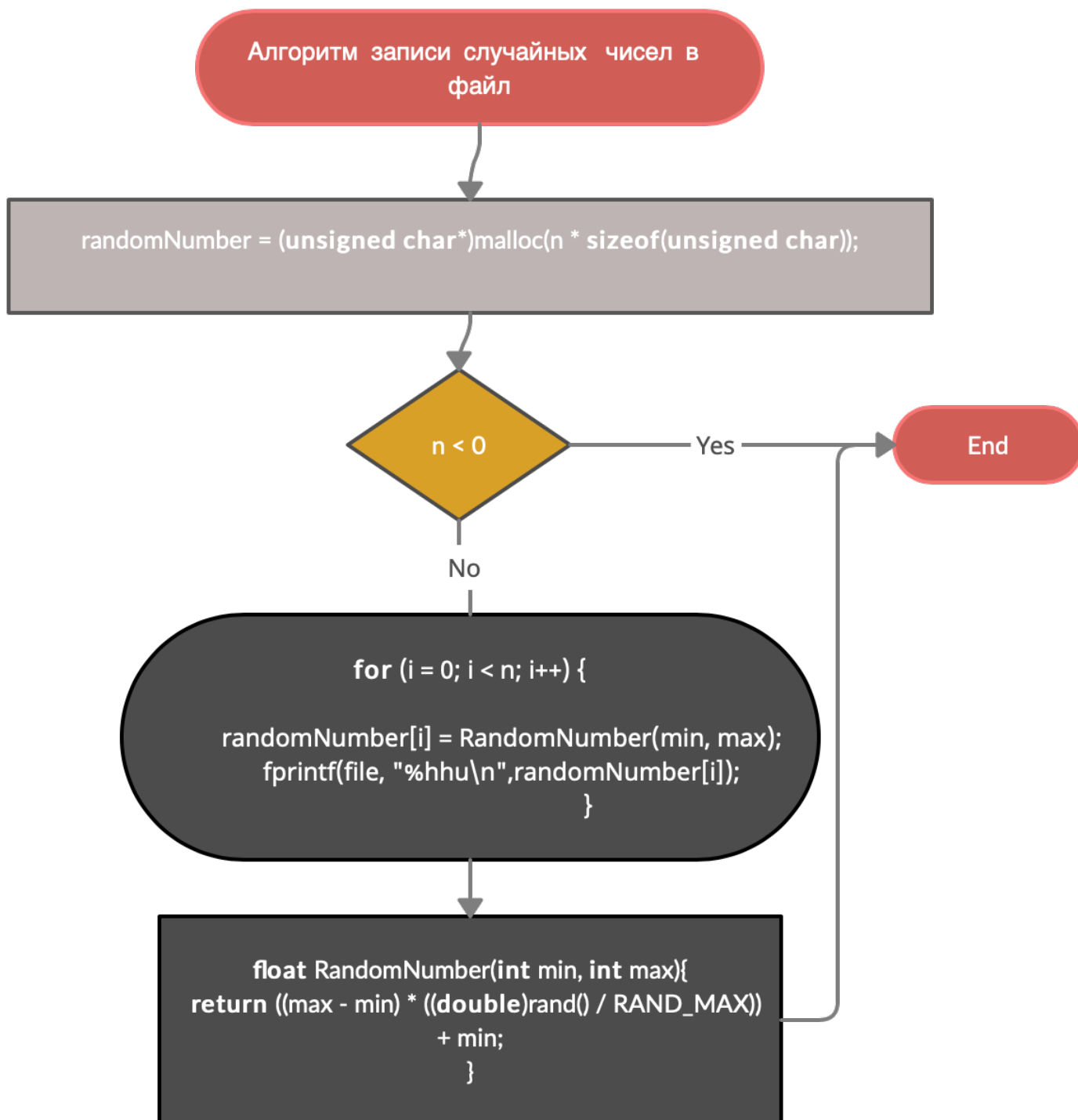
```

5. Функция сортировок с таймером.

```
void fastSort(int linesCount, unsigned char* randomNumber, double
t, FILE* fp){
    rewind(fp);
    clock_t start = clock();
    quickSort(randomNumber, 0, linesCount - 1);
    clock_t end = clock();
    t = ((double)(end - start)) / CLOCKS_PER_SEC;
    printf("Время, занятое сортировкой: %lf\n", t);
}

void insertSort(int linesCount, unsigned char* randomNumber,
double t, FILE* fp){
    rewind(fp);
    clock_t start = clock();
    InsertionSort(linesCount, randomNumber);
    clock_t end = clock();
    t = ((double)(end - start)) / CLOCKS_PER_SEC;
    printf("Время отсортировки: %lf\n", t);
}

void bubbleSort(int linesCount, unsigned char* randomNumber,
double t, FILE* fp){
    rewind(fp);
    clock_t start = clock();
    BubbleSort(linesCount, randomNumber);
    clock_t end = clock();
    t = ((double)(end - start)) / CLOCKS_PER_SEC;
    printf("Время отсортировки: %lf\n", t);
}
```



Алгоритм считывания случайных чисел из файла

```
fp = fopen("textFile.txt", "r");  
  
while((ch=fgetc(fp))!=EOF) {  
    if(ch=='\n')  
        linesCount++;  
}
```

```
randomNumber = (unsigned char*)malloc(linesCount * sizeof(unsigned char));
```

```
rewind(fp);  
for (i = 0; i < linesCount; i++){  
    fscanf(fp, "%hhu", &randomNumber[i]);  
}
```

End

Эксперименты.

Для начала убедимся, что первая программа записывает числа в файл. Изначально «Блокнот» чист:

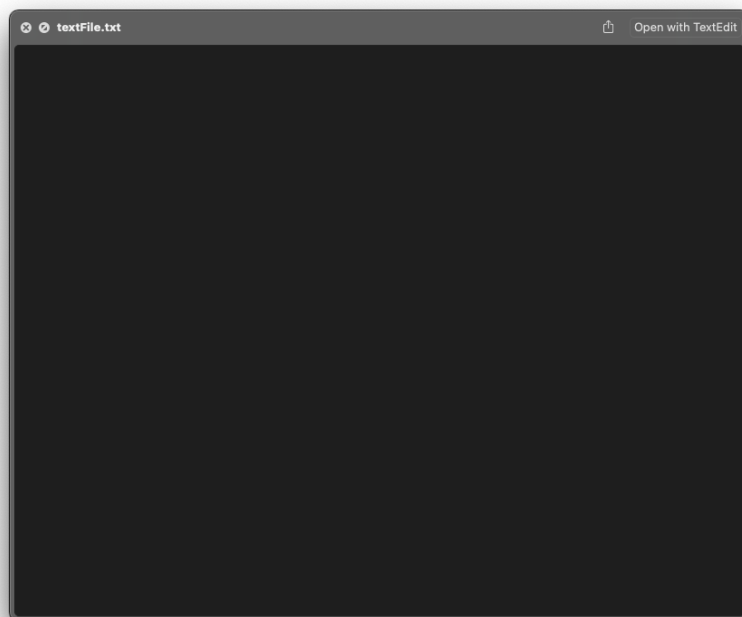


Рисунок 9. Чистый текстовый файл.

Далее запустим программу для записи чисел в файл и проверим его на наличие записанных чисел:

```
Выберите действие:
1) Ввести с клавиатуры
2) Сгенерировать случайные числа
3) Завершить работу
Ваш выбор: 2
Введите кол-во случайных чисел: 100
Введите минимум диапазона: -999
Введите максимум диапазона: 999

Полученная сумма: 11101.000000
```

Рисунок 10. Выполнение программы.

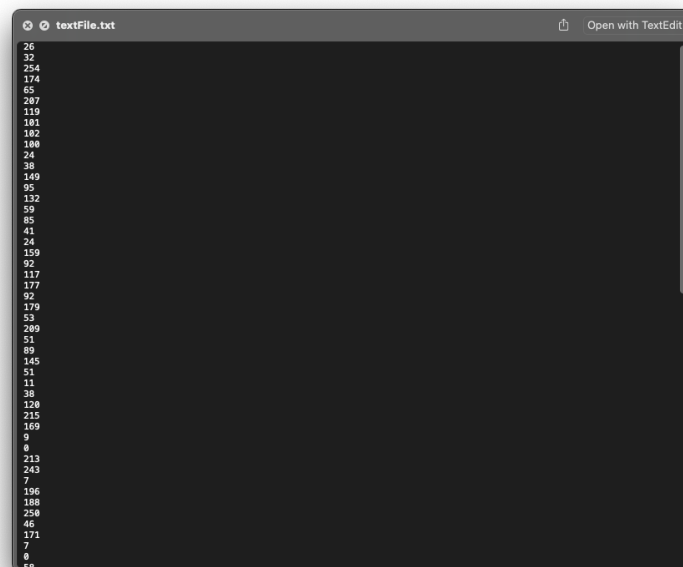


Рисунок 11. Записанные числа в файл.

Действительно, программа записала 100 случайных чисел в правильном диапазоне.

Далее проверим правильность выполнения второй программы, которая представляет собой консольный интерфейс.

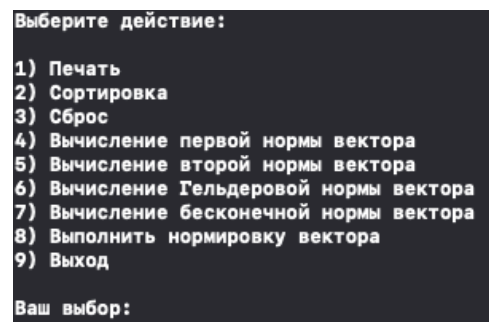


Рисунок 12. Консольный интерфейс.

Выберем команду «Печать». Для этого введем «1» и нажмем «Enter». На экран будут выведены числа, записанные в файле:

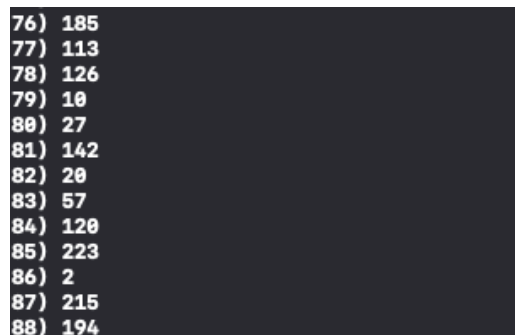


Рисунок 13. Печать.

Теперь отсортируем их методом пузырька и выведем занятое время:

```
Выберите метод сортировки:
1. Пузырьком
2. Вставкой
3. Быстрая

Ваш выбор: 1
Время отсортировки: 0.000061
```

Рисунок 14. Сортировка пузырьком и вывод результата.

Проверим сброс сортировки. Выберем команду «Сброс», после «Печать»:

```
Выберите действие:
1) Печать
2) Сортировка
3) Сброс
4) Вычисление первой нормы вектора
5) Вычисление второй нормы вектора
6) Вычисление Гельдеровой нормы вектора
7) Вычисление бесконечной нормы вектора
8) Выполнить нормировку вектора
9) Выход

Ваш выбор: 1
1) 26
2) 32
3) 254
4) 174
5) 65
6) 207
7) 119
8) 101
9) 102
10) 100
11) 24
12) 38
13) 149
14) 95
15) 132
16) 59
```

Рисунок 15. Сброс сортировки.

Проверим оставшиеся две сортировки. Для этого после каждой сортировки будем сбрасывать ее:

```
Выберите метод сортировки:
1. Пузырьком
2. Вставкой
3. Быстрая

Ваш выбор: 2
Время отсортировки: 0.000019
```

Рисунок 16. Сортировка вставками и вывод результата

```
Ваш выбор: Выберите метод сортировки:
1. Пузырьком
2. Вставкой
3. Быстрая

Ваш выбор: 3
Время, занятое сортировкой: 0.000020
```

Рисунок 17. Быстрая сортировка и вывод результата

Проверим программу на введение некорректного номера команды. Для этого введем номер, не присвоенный ни одной из команд:

```
Выберите действие:
1) Печать
2) Сортировка
3) Сброс
4) Вычисление первой нормы вектора
5) Вычисление второй нормы вектора
6) Вычисление Гельдеровой нормы вектора
7) Вычисление бесконечной нормы вектора
8) Выполнить нормировку вектора
9) Выход

Ваш выбор: 10
Не корректный ввод
```

Рисунок 18. Некорректный номер команды.

Наконец, выберем команду «Выход» и завершим программу. На экран выведется сообщение о завершении.

```

Выберите действие:
1) Печать
2) Сортировка
3) Сброс
4) Вычисление первой нормы вектора
5) Вычисление второй нормы вектора
6) Вычисление Гельдеровой нормы вектора
7) Вычисление бесконечной нормы вектора
8) Выполнить нормировку вектора
9) Выход

Ваш выбор: 9
Завершение программы...

Program ended with exit code: 0|

```

Рисунок 19. Завершение работы программы.

Тип сортировки	Сложность	Размер входных данных	Время работы, с
Пузырьком	$O(n^2)$	100000	$\approx 30,024$
Вставками	$O(n^2)$	100000	$\approx 6,316$
Быстрая	$O(n \log n)$	100000	$\approx 0,014$
Пузырьком	$O(n^2)$	500000	$\approx 839,196$
Вставками	$O(n^2)$	500000	$\approx 180,497$
Быстрая	$O(n \log n)$	500000	$\approx 0,074$

Заключение

В ходе лабораторной работы была написана программа на языке программирования «С», которая полностью выполняет поставленную задачу, а именно:

«Сравнение сортировок.

Реализовать сортировки массивов данных (тип данных “unsigned char”) задаваемых: обязательно случайно, дополнительно с клавиатуры или из файла.

Реализовать сортировки: пузырьком, вставкой, быстрая.

Реализовать вычисление и нормировку вектора.

Сравнить время работы, сделать выводы.

Первая программа создает текстовый файл с записанными в него числами. Программа принимает количество чисел n , максимальное и минимальное значение.

Вторая программа читает текстовый файл с набором чисел, выводит консольный интерфейс (печать, сортировка, сброс, Вычисление первой нормы вектора, Вычисление второй нормы вектора, Вычисление Гельдеревой нормы вектора, Вычисление бесконечной нормы вектора, Выполнить нормировку вектора, выход), выполняет выбранные действия».

Интерфейс программы простой, понятный и удобный в использовании, все сообщения выводятся на русском языке и при своих небольших размерах содержат нужную информацию, которую необходимо донести до пользователя.

В ходе сравнения сортировок можно сделать вывод: «быстрая сортировка (quick sort)» выполняет сортировку чисел быстрее остальных типов сортировки и очень быстро работает с любыми типами данных; сортировка «пузырьком» занимает большее время чем «quick sort», но тоже довольно быстро сортирует данные; «сортировка вставками» работает медленнее всех на большом диапазоне чисел, ее следует использовать на коротком отрезке не больше, чем десятки, или уже на частично отсортированном массиве данных.

Список литературы

1. Лекции Сысоева Александра Владимировича, 2021.
2. Практики и лабораторные Лебедева Ильи Геннадьевича, 2021.

Приложение 1 (Генерирует числа)

```
//
//  main.c
//  Laboratorka
//
//  Created by Kamal Muradov on 27.10.2021.
//

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

float RandomFloat(int min, int max){
    return ((max - min) * ((double)rand() / RAND_MAX)) + min;
}

int main(){

    FILE *file;

    int n = 0, min = 0, max = 0, i = 0, indexOfNumber = 1;
    double sum = 0.0;
    unsigned char *randomNumber, *keyboardNumber;
    int menuOption = 0;
    int* randomNumberDecimal;

    do{
        printf("Выберите действие:\n\n1) Ввести с клавиатуры\n2) Сгенерировать случайные числа\n3) Завершить работу\nВаш выбор: ");
        scanf("%d", &menuOption);
        switch(menuOption){
            case 1:
                file = fopen("textFile.txt", "w");
                n = 0;
                printf("Выбрано действие 'Ввести с клавиатуры...\n");
                printf("Введите количество вводимых чисел: ");
                scanf("%d", &n);
                if(n <= 0) {
                    printf("Кол-во чисел не может быть меньше или равно 0.\n");
                    menuOption = 1;
                    break;
                } else{
                    keyboardNumber = (unsigned char*)malloc(n * sizeof(unsigned char));
                    printf("Вводимое число не может быть меньше 0 и больше 255\n");

                    for (i = 1; i < n + 1; i++){
```

```

scanf("%hhu", keyboardNumber);
fprintf ("%hhu\n%hhu", *keyboardNumber);
}
printf("\nИдет запись чисел. Ожидание...\n\n");
sleep(1);
printf("Запись чисел с клавиатуры была
произведена успешно!");

}
printf("Запись прошла успешно!\n");
fclose(file);
free(keyboardNumber);
break;
case 2:
file = fopen("textFile.txt", "w");
n = 0;
printf("Введите кол-во случайных чисел: ");
scanf("%d", &n);

printf("Введите минимум диапазона: ");
scanf("%d", &min);

printf("Введите максимум диапазона: ");
scanf("%d", &max);

printf("\n");

randomNumber = (unsigned char*)malloc(n *
sizeof(unsigned char));
randomNumberDecimal = (int*)malloc(n * sizeof(int));

// создаем случайное число и узнаем десятичную часть
числа
for (i = 0; i < n; i++) {

    randomNumber[i] = RandomFloat(min, max);
    randomNumberDecimal[i] = (randomNumber[i] -
(int)randomNumber[i]) * 1000000;
    fprintf(file, "%hhu\n", randomNumber[i]);

}
// проверяем условия для суммирования
for (i = 0; i < n; i++){

    if (randomNumberDecimal[i] == indexOfNumber) {
        sum -= randomNumber[i];
    }
    else {
        sum += randomNumber[i];
    }
    indexOfNumber++;
}
// вывод суммы

```



```

        printf("Полученная сумма: %f\n\n", sum);
        free(randomNumber);
        free(randomNumberDecimal);
        fclose(file);
        break;
    case 3:
        printf("Завершение работы.\n\n");
        sleep(3);
    }
}while(menuOption != 3);

return 0; // программа успешно завершена
}

```

Приложение 2 (Сортировка чисел)

```

//
//  second.c
//  laba2
//
//  Created by Kamal Muradov on 28.11.2021.
//

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <time.h>
#include <math.h>
#include "sortingsHeader.h"

void fastSort(int linesCount, unsigned char* randomNumber, double t,
FILE* fp){
    rewind(fp);
    clock_t start = clock();
    quickSort(randomNumber, 0, linesCount - 1);
    clock_t end = clock();
    t = ((double)(end - start)) / CLOCKS_PER_SEC;
    printf("Время, занятое сортировкой: %lf\n", t);
}

void insertSort(int linesCount, unsigned char* randomNumber, double t,
FILE* fp){
    rewind(fp);
}

```

```

    clock_t start = clock();
    InsertionSort(linesCount, randomNumber);
    clock_t end = clock();
    t = ((double)(end - start)) / CLOCKS_PER_SEC;
    printf("Время отсортировки: %lf\n", t);
}

void bubbleSort(int linesCount, unsigned char* randomNumber, double t,
FILE* fp){
    rewind(fp);
    clock_t start = clock();
    BubbleSort(linesCount, randomNumber);
    clock_t end = clock();
    t = ((double)(end - start)) / CLOCKS_PER_SEC;
    printf("Время отсортировки: %lf\n", t);
}

int main(int argc, char *argv[]){

    FILE* fp;
    char ch;
    int i, sort = 0, p = 0;
    int linesCount = 0;
    double secondSum = 0, sqrtedNums = 0, maxVector = -INFINITY,
normVector = 0;
    unsigned char* randomNumber, *pRandomNumber;
    double t = 0;
    int menuOption = 0;

    fp = fopen("textFile.txt", "r");

    while((ch=fgetc(fp))!=EOF) {
        if(ch=='\n')
            linesCount++;
    }

    randomNumber = (unsigned char*)malloc(linesCount * sizeof(unsigned
char));
    pRandomNumber = (unsigned char*)malloc(linesCount * sizeof(unsigned
char));

    rewind(fp);
    for (i = 0; i < linesCount; i++){
        fscanf(fp, "%hu", &randomNumber[i]);
    }

    if(argc == 2){
        if (strcmp("--fast", argv[1]) == 0) {
            fastSort(linesCount,randomNumber,t,fp);
            for (i = 0; i < linesCount; i++){
                printf("%d) %hu\n", i + 1, randomNumber[i]);
            }
        }
    }
}

```

```

    }
    return 0;
} else if (strcmp("--bubble", argv[1]) == 0) {
    bubbleSort(linesCount, randomNumber, t, fp);
    for (i = 0; i < linesCount; i++){
        printf("%d) %hhu\n", i + 1, randomNumber[i]);
    }
    return 0;
} else if (strcmp("--insertion", argv[1]) == 0) {
    insertSort(linesCount, randomNumber, t, fp);
    for (i = 0; i < linesCount; i++){
        printf("%d) %hhu\n", i + 1, randomNumber[i]);
    }
    return 0;
}
}

do{
    printf("Выберите действие:\n\n1) Печать\n2) Сортировка\n3)
Сброс\n4) Вычисление первой нормы вектора\n5) Вычисление второй нормы
вектора\n6) Вычисление Гельдеровой нормы вектора\n7) Вычисление
бесконечной нормы вектора\n8) Выполнить нормировку вектора\n9)
Выход\n\nВаш выбор: ");
    scanf("%d", &menuOption);
    secondSum = 0;

    switch(menuOption){
        case 1:
            for (i = 0; i < linesCount; i++)
            {
                printf("%d) %hhu\n", i + 1, randomNumber[i]);
            }
            sleep(2);
            break;

        case 2:
            do{
                printf("Выберите метод сортировки:\n1. Пузырьком\n2.
Вставкой\n3. Быстрая\n\nВаш выбор: ");
                scanf("%d", &sort);
                switch(sort){
                    case 1: // пузырьк
                        bubbleSort(linesCount, randomNumber, t, fp);
                        break;
                    case 2: // вставка
                        insertSort(linesCount, randomNumber, t, fp);
                        break;
                    case 3:
                        fastSort(linesCount, randomNumber, t, fp);
                        break;
                    default:

```

```

        printf("Не правильный выбор, пожалуйста
попробуйте еще раз.\n\n");
        break;
    }
    }while(sort <= 0 || sort > 3);

    break;

case 3:
    for (i = 0; i < linesCount; i++){
        fscanf(fp, "%hhu", &randomNumber[i]);
    }
    printf("Массив сброшен.\n");
    sleep(2);
    break;

case 4:
    for(i = 0; i < linesCount; i++){
        pRandomNumber[i] = (randomNumber[i]);
        secondSum += pRandomNumber[i];
    }
    printf("%f\n", secondSum);
    sleep(2);
    break;

case 5:
    for(i = 0; i < linesCount; i++){
        pRandomNumber[i] = (randomNumber[i]);
        secondSum += pRandomNumber[i] * pRandomNumber[i];
    }

    printf("%f\n", sqrt(secondSum));
    sleep(2);
    break;

case 6:
    printf("Введите значение P: ");
    scanf("%d", &p);
    for(i = 0; i < linesCount; i++){
        secondSum += pow((randomNumber[i]), p);
    }
    normVector = pow(secondSum, 1.0/p);
    printf("%f\n\n", normVector);
    sleep(2);
    break;

case 7:
    for(i = 0; i < linesCount; i++){
        pRandomNumber[i] = (randomNumber[i]);
    }
    for(i = 0; i < linesCount; i++){

```

```

        if(pRandomNumber[i] > maxVector){
            maxVector = pRandomNumber[i];
        }
    }
    printf("Бесконечная норма вектора равна %f", maxVector);
    sleep(2);
    break;

    case 8:
        for(i = 0; i < linesCount; i++){
            pRandomNumber[i] = (randomNumber[i]);
            secondSum += pRandomNumber[i] * pRandomNumber[i];
        }
        sqrtedNums = sqrt(secondSum);
        for(i = 0; i < linesCount; i++){
            printf("%d) %f\n", i + 1, randomNumber[i]/sqrtedNums);
        }
        sleep(2);
        break;

    case 9:
        printf("Завершение программы...\n\n");
        sleep(3);
        break;

    default:
        printf("Не корректный ввод\n\n");
}

}while(menuOption != 9);

fclose(fp);
free(randomNumber);

return 0;
}

```

Приложение 3 (Функции Сортировок)

```

#include <stdio.h>
#include <stdlib.h>

//сортировка пузырьком
void BubbleSort(int n, unsigned char a[]) {

    int i, j;

    for (i = 0; i < n - 1; i++)
    {

```

```

    for (j = 0; j < n - i - 1; j++)
    {
        if (a[j] > a[j + 1])
        {
            double tmp = a[j];
            a[j] = a[j + 1];
            a[j + 1] = tmp;
        }
    }
}

//сортировка вставкой
void InsertionSort(int n, unsigned char a[]) {

    int i;
    int location;
    double newElement;

    for (i = 1; i < n; i++)
    {
        newElement = a[i];
        location = i - 1;
        while (location >= 0 && a[location] > newElement)
        {
            a[location + 1] = a[location];
            location = location - 1;
        }
        a[location + 1] = newElement;
    }
}

//быстрая сортировка
void quickSort(unsigned char* a, int left, int right) {

    int i, j;
    double x, y;

    i = left;
    j = right;
    x = a[(left + right) / 2];
    do
    {
        while ((a[i] < x) && (i < right))
            i++;
        while ((x < a[j]) && (j > left))
            j--;
        if (i <= j)
        {
            y = a[i];
            a[i] = a[j];
            a[j] = y;
        }
    }
}

```

```

        i++;
        j--;
    }
} while (i <= j);
if (left < j){
    quickSort(a, left, j);

}
if (i < right){
    quickSort(a, i, right);

}
}
}

```

Приложение 4 (Головной файл)

```

//
//  sortingsHeader.h
//  laba2
//
//  Created by Kamal Muradov on 28.11.2021.
//

#ifndef _HEADER_H
#define _HEADER_H

void BubbleSort(int n, unsigned char a[]);

void InsertionSort(int n, unsigned char a[]);

void quickSort(unsigned char* items, int left, int right);

#endif // !_HEADER_H

```